

Nokia Corporation Docket No.: NC35320
Harrington & Smith, LLP Docket No.: 872.0111.U1(US)
Patent Application Papers of: Petri Laukkanen
Minna-Mari Savolainen

**METHOD AND APPARATUS FOR PROVIDING HINDI INPUT
TO A DEVICE USING A NUMERIC KEYPAD**

10027445.43001
FOOTNOTES

| Table 1. Demographic characteristics of the study population | |
|--|----------------|
| Age (years) | 65.0 ± 10.0 |
| Gender (male/female) | 100/100 |
| Education (years) | 12.0 ± 2.0 |
| Occupation (white/blue) | 100/100 |
| Marital status (married/divorced/widowed) | 100/100/100 |
| Income (USD/month) | 1000.0 ± 200.0 |
| Health status (good/poor) | 100/100 |
| Smoking status (smoker/non-smoker) | 100/100 |
| Alcohol consumption (yes/no) | 100/100 |
| Comorbidities (hypertension/diabetes/cholesterol) | 100/100/100 |
| Medication (yes/no) | 100/100 |
| Family history (yes/no) | 100/100 |
| Genetic testing (yes/no) | 100/100 |
| Genetic testing results (normal/abnormal) | 100/100 |
| Genetic testing cost (USD) | 100.0 ± 20.0 |
| Genetic testing time (hours) | 1.0 ± 0.5 |
| Genetic testing accuracy (%) | 95.0 ± 2.0 |
| Genetic testing sensitivity (%) | 90.0 ± 3.0 |
| Genetic testing specificity (%) | 98.0 ± 1.0 |
| Genetic testing reliability (%) | 99.0 ± 0.5 |
| Genetic testing validity (%) | 97.0 ± 1.5 |
| Genetic testing robustness (%) | 96.0 ± 2.0 |
| Genetic testing precision (%) | 94.0 ± 2.5 |
| Genetic testing repeatability (%) | 93.0 ± 3.0 |
| Genetic testing reproducibility (%) | 92.0 ± 3.5 |
| Genetic testing consistency (%) | 91.0 ± 4.0 |
| Genetic testing stability (%) | 90.0 ± 4.5 |
| Genetic testing durability (%) | 89.0 ± 5.0 |
| Genetic testing portability (%) | 88.0 ± 5.5 |
| Genetic testing scalability (%) | 87.0 ± 6.0 |
| Genetic testing interoperability (%) | 86.0 ± 6.5 |
| Genetic testing compatibility (%) | 85.0 ± 7.0 |
| Genetic testing accessibility (%) | 84.0 ± 7.5 |
| Genetic testing availability (%) | 83.0 ± 8.0 |
| Genetic testing reliability (%) | 82.0 ± 8.5 |
| Genetic testing validity (%) | 81.0 ± 9.0 |
| Genetic testing robustness (%) | 80.0 ± 9.5 |
| Genetic testing precision (%) | 79.0 ± 10.0 |
| Genetic testing repeatability (%) | 78.0 ± 10.5 |
| Genetic testing reproducibility (%) | 77.0 ± 11.0 |
| Genetic testing consistency (%) | 76.0 ± 11.5 |
| Genetic testing stability (%) | 75.0 ± 12.0 |
| Genetic testing durability (%) | 74.0 ± 12.5 |
| Genetic testing portability (%) | 73.0 ± 13.0 |
| Genetic testing scalability (%) | 72.0 ± 13.5 |
| Genetic testing interoperability (%) | 71.0 ± 14.0 |
| Genetic testing compatibility (%) | 70.0 ± 14.5 |
| Genetic testing accessibility (%) | 69.0 ± 15.0 |
| Genetic testing availability (%) | 68.0 ± 15.5 |
| Genetic testing reliability (%) | 67.0 ± 16.0 |
| Genetic testing validity (%) | 66.0 ± 16.5 |
| Genetic testing robustness (%) | 65.0 ± 17.0 |
| Genetic testing precision (%) | 64.0 ± 17.5 |
| Genetic testing repeatability (%) | 63.0 ± 18.0 |
| Genetic testing reproducibility (%) | 62.0 ± 18.5 |
| Genetic testing consistency (%) | 61.0 ± 19.0 |
| Genetic testing stability (%) | 60.0 ± 19.5 |
| Genetic testing durability (%) | 59.0 ± 20.0 |
| Genetic testing portability (%) | 58.0 ± 20.5 |
| Genetic testing scalability (%) | 57.0 ± 21.0 |
| Genetic testing interoperability (%) | 56.0 ± 21.5 |
| Genetic testing compatibility (%) | 55.0 ± 22.0 |
| Genetic testing accessibility (%) | 54.0 ± 22.5 |
| Genetic testing availability (%) | 53.0 ± 23.0 |
| Genetic testing reliability (%) | 52.0 ± 23.5 |
| Genetic testing validity (%) | 51.0 ± 24.0 |
| Genetic testing robustness (%) | 50.0 ± 24.5 |
| Genetic testing precision (%) | 49.0 ± 25.0 |
| Genetic testing repeatability (%) | 48.0 ± 25.5 |
| Genetic testing reproducibility (%) | 47.0 ± 26.0 |
| Genetic testing consistency (%) | 46.0 ± 26.5 |
| Genetic testing stability (%) | 45.0 ± 27.0 |
| Genetic testing durability (%) | 44.0 ± 27.5 |
| Genetic testing portability (%) | 43.0 ± 28.0 |
| Genetic testing scalability (%) | 42.0 ± 28.5 |
| Genetic testing interoperability (%) | 41.0 ± 29.0 |
| Genetic testing compatibility (%) | 40.0 ± 29.5 |
| Genetic testing accessibility (%) | 39.0 ± 30.0 |
| Genetic testing availability (%) | 38.0 ± 30.5 |
| Genetic testing reliability (%) | 37.0 ± 31.0 |
| Genetic testing validity (%) | 36.0 ± 31.5 |
| Genetic testing robustness (%) | 35.0 ± 32.0 |
| Genetic testing precision (%) | 34.0 ± 32.5 |
| Genetic testing repeatability (%) | 33.0 ± 33.0 |
| Genetic testing reproducibility (%) | 32.0 ± 33.5 |
| Genetic testing consistency (%) | 31.0 ± 34.0 |
| Genetic testing stability (%) | 30.0 ± 34.5 |
| Genetic testing durability (%) | 29.0 ± 35.0 |
| Genetic testing portability (%) | 28.0 ± 35.5 |
| Genetic testing scalability (%) | 27.0 ± 36.0 |
| Genetic testing interoperability (%) | 26.0 ± 36.5 |
| Genetic testing compatibility (%) | 25.0 ± 37.0 |
| Genetic testing accessibility (%) | 24.0 ± 37.5 |
| Genetic testing availability (%) | 23.0 ± 38.0 |
| Genetic testing reliability (%) | 22.0 ± 38.5 |
| Genetic testing validity (%) | 21.0 ± 39.0 |
| Genetic testing robustness (%) | 20.0 ± 39.5 |
| Genetic testing precision (%) | 19.0 ± 40.0 |
| Genetic testing repeatability (%) | 18.0 ± 40.5 |
| Genetic testing reproducibility (%) | 17.0 ± 41.0 |
| Genetic testing consistency (%) | 16.0 ± 41.5 |
| Genetic testing stability (%) | 15.0 ± 42.0 |
| Genetic testing durability (%) | 14.0 ± 42.5 |
| Genetic testing portability (%) | 13.0 ± 43.0 |
| Genetic testing scalability (%) | 12.0 ± 43.5 |
| Genetic testing interoperability (%) | 11.0 ± 44.0 |
| Genetic testing compatibility (%) | 10.0 ± 44.5 |
| Genetic testing accessibility (%) | 9.0 ± 45.0 |
| Genetic testing availability (%) | 8.0 ± 45.5 |
| Genetic testing reliability (%) | 7.0 ± 46.0 |
| Genetic testing validity (%) | 6.0 ± 46.5 |
| Genetic testing robustness (%) | 5.0 ± 47.0 |
| Genetic testing precision (%) | 4.0 ± 47.5 |
| Genetic testing repeatability (%) | 3.0 ± 48.0 |
| Genetic testing reproducibility (%) | 2.0 ± 48.5 |
| Genetic testing consistency (%) | 1.0 ± 49.0 |
| Genetic testing stability (%) | 0.0 ± 49.5 |
| Genetic testing durability (%) | -1.0 ± 50.0 |
| Genetic testing portability (%) | -2.0 ± 50.5 |

5

10

15

20

25

While well suited for use with those languages that employ the Latin alphabet, this conventional technique can prove to be limiting when attempting to enter other types of alphabetic symbols associated with other languages, which can have more complex rules relating to the relationships

between the constituent parts of the written language, such as vowels, consonants and special characters and modifiers. Hindi is one such language.

As such, to enter a language such as Hindi a significant problem arises when one desires to
 5 employ a device capable of text entry, such as a cellular telephone, that is provided only with a simple 12 key numeric-type of keypad.

ISCII (IS 13194:1991 Indian Script Code for Information Interchange) specifies an alphabetic code that can be used for encoding multiple Indian national languages with a single codeset.
 10 ISCII has been used as the basis for the Indian script code point allocation in Unicode (Devanagari, in particular, is allocated in U+0900-U+097F in Unicode). In its annex D, the standard specifies an INSCRIPT keyboard for a PC environment.

INSCRIPT specifies a phonetic PC keyboard for Indian languages. Similarly to the general
 15 design of ISCII, the INSCRIPT keyboard is designed to work as a solution for multiple Indian languages. It provides a phonetic mapping of the keyboard that is readily usable across the Indian languages, even though they use distinct scripts for the writing of the language.

The words of the Indian language are typed in and stored in the strict phonetic order, even if the
 20 phonetic order may not always correspond to the visual appearance of the displayed text.

The INSCRIPT keyboard requires two distinct keys for each pair of independent and dependent vowels. The INSCRIPT also requires the explicit use of virama (vowel killer) to produce consonant clusters. As may be appreciated, this approach does not lend itself for implementation
 25 on a limited keypad type of user input device, such as an ITU-T keypad of a type that is typically found in a cellular telephone.

SUMMARY OF THE PREFERRED EMBODIMENTS

30 The foregoing and other problems are overcome, and other advantages are realized, in accordance with the presently preferred embodiments of these teachings.

As was stated, the INSCRIPT keyboard requires two distinct keys for each pair of independent and dependent vowel, whereas this invention requires only one key for such pair. The distinction between independent and dependent vowels is either made automatically (intelligence based on the current editing context), or it is made explicitly by using a predetermined context shift key.

5

As was also noted above, the INSCRIPT approach requires the explicit use of virama (vowel killer) to produce consonant clusters, whereas the invention hides the use of virama from the user, and consonants are combined into clusters by using the context shift key. In practice, the user simply types in the text using a procedure similar to the multi-tapping that is widely used for entering Latin/Cyrillic/Greek/Arabic/Hebrew scripts. In most cases the user can rely on the default behavior of an editor function to make the proper choices, and when it becomes necessary to override the default behavior the user simply depresses the context shift key.

10

A method is disclosed for operating a device having a numeric keypad to provide intelligent text input for characters. While described below in the context of Hindi (Devanagari script), these teachings can be used as well for other Indian scripts, including Punjabi, Gujarati, Oriya, Bengali, Assamese, Telugu, Kannada, Malayalam and Tamil.

15

In this invention the keypad keys are redefined to be vowel keys, consonant keys, and one or more other keys, such as a context switch key. As but one non-limiting example, in this method predetermined number keys, such as 1 and 2 (or 2 and 3), are reserved for vowels. Depressing a vowel key inserts either an independent vowel or a dependent vowel (matra) in a text buffer, depending on the character preceding the current character insertion point. If the character preceding the insertion point is a consonant (of the relevant script being edited), depressing a vowel key produces a dependent vowel, in all other cases depressing a vowel key produces an independent vowel. Note in this regard that when inputting Hindi, if the insertion point happens to be preceded by a consonant from a language or script other than Hindi, e.g., the insertion point is preceded by a Latin (English) consonant "n", pressing a vowel key produces an independent vowel rather than a dependent vowel, as the preceding character is not a consonant of the relevant script (Hindi/Devanagari).

25

30

If the user wishes to change the dependent vowel into an independent vowel, the user can do this

by pressing the context switch key, which may be the * key, and the vowel key in sequence. As a result, the vowel is inserted into the text buffer as an independent vowel, thereby overriding the context into which the vowel is being entered.

- 5 As but one further non-limiting example, in this method predetermined number keys, such as 3,4,5,6,7,8 and 9 (or 4, 5, 6, 7, 8 and 9), are reserved for the consonants. If the user presses any of the consonant keys, a consonant is inserted into the text buffer and displayed.

The user may enter a consonant cluster by activating the context shift key to change a previously
10 entered consonant into a half-form, and then entering a second consonant. A nukta consonant can be entered by entering a consonant, followed by selecting a nukta modifier.

In accordance with this invention all of the alphabetic characters of an Indian script are mapped to the numeric keypad of a mobile telephone. For example, in accordance with an embodiment
15 suitable for entering Hindi characters the mapping of the numeric keypad is as follows: key 0 is a space key; key 1 has modifiers mapped to it; keys 2-3 have vowels mapped to them; and keys 4-9 have consonants mapped to them. Other script-specific characters, such as punctuation, can be accessed from a special character list that may be accessed in any one of a number of suitable ways. In addition to the numeric keys of the numeric keypad, the input method designed for the input of Indian
20 languages includes a context shift key, preferably but not necessarily mapped to the key * of the ITU-T keypad.

The vowel keys 2-3 have a dual function: depending on the current editing context they either produce independent vowels (Unicode reference U+0905-U+0914) or matras, i.e., dependent vowels (Unicode
25 reference U+093E-U+094C). The default operation can be overridden by pressing the context shift key prior to pressing a vowel key.

The consonant keys 4-9 normally produce live or full consonants. The default operation can be overridden by pressing the context shift key prior to pressing a consonant key.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other aspects of these teachings are made more evident in the following Detailed Description of the Preferred Embodiments, when read in conjunction with the attached
 5 Drawing Figures, wherein:

Fig. 1 is block diagram of a mobile station having a numeric keypad that is suitably programmed for implementing the teachings of this invention;

10 Fig. 2 shows Hindi character matras, or dependent vowels;

Fig. 3 shows a halant;

Figs. 4A, 4B and 4C show modifiers, specifically a candrabindu, inserted above a character (Fig. 4A), an anusvara, inserted above a character (Fig. 4B), and a visarga, inserted beside a character
 15 (Fig. 4C);

Fig. 5 shows a nukta, that functions as a combining mark;

20 Fig. 6 shows a mapping of Hindi vowels and consonants relative to numeric keypad keys;

Fig. 7 illustrates how multiple key depressions are used to scroll through the Hindi characters that are mapped behind a specific key (the 3 key in this case);

25 Fig. 8 shows the mapping of the Hindi keypad, and specifically the mapping of matras (dependent vowels) on keys 1 and 2, and consonants on keys 3-9 (the zero 0 key is used to enter whitespace);

Fig. 9 shows the mapping of the Hindi keypad, and specifically the mapping of independent
 30 vowels on keys 1 and 2, and the consonants on keys 3-9;

Fig. 10 shows an example of sample text, including the preferred use of Latin numbers as opposed to Devanagari script numbers, and allowable cursor stopping points between Hindi syllables; and

5 Fig. 11 shows an example of the addition and the subsequent deletion of characters.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring first to Fig. 1, there is illustrated a simplified block diagram of an embodiment of a
 10 wireless communications system 5 that includes at least one mobile station (MS) 100 suitable for practicing this invention. Fig. 1 also shows an exemplary network operator having, for example, a GPRS Support Node (GSN) 30 for connecting to a telecommunications network, such as a Public Packet Data Network or PDN, at least one base station controller (BSC) 40, and a plurality of base transceiver stations (BTS) 50 that transmit in a forward or downlink direction
 15 both physical and logical channels to the mobile station 100 in accordance with a predetermined air interface standard. A reverse or uplink communication path also exists from the mobile station 100 to the network operator, which conveys mobile originated access requests and traffic.

The air interface standard can conform to any suitable standard or protocol, and may enable both
 20 voice and data traffic, such as data traffic enabling Internet 70 access and web page downloads. The air interface standard could be a Time Division Multiple Access (TDMA) air interface that supports a GSM or an advanced GSM protocol and air interface, or it could be a Code Division Multiple Access (CDMA) air interface, including a wideband CDMA (WCDMA) air interface. In some embodiments the air interface could as well be one similar to or based on a conventional
 25 analog air interface, such as one known in North America as AMPS, if there is some utility to be derived from the user entering text into the mobile station 100.

The network operator may also include a suitable type of Message Center (MC) 60 that receives and forwards messages for the mobile stations 100. Other types of messaging service may
 30 include Supplementary Data Services and one under currently development and known as Multimedia Messaging Service (MMS), wherein image messages, video messages, audio messages, text messages, executables and the like, and combinations thereof, can be transferred

between the network and the mobile station 100.

The mobile station 100 may be a handheld radiotelephone, such as a cellular telephone or a personal communicator, or it could be a component of an electronic organizer type of device.

- 5 The mobile station 100 could also be contained within a card or module that is connected during use to another device. For example, the mobile station 10 could be contained within a PCMCIA or similar type of card or module that is installed during use within a portable data processor, such as a laptop or notebook computer. In these cases it is assumed that a keypad of the host device is operated by the user to enter Hindi text into the mobile station 100, as will be described
- 10 in detail below. The host computer could be one that is wearable by the user.

The mobile station 100 typically includes a data processor, typically embodied as a microcontrol unit (MCU) 120 having an output coupled to an input of a display 140 and an input coupled to an output of a keyboard or keypad 160.

15

The MCU 120 is assumed to include or be coupled to some type of a memory 130, including a read-only memory (ROM) for storing an operating program, as well as a random access memory (RAM) for temporarily storing required data, scratchpad memory, messages being composed and those that have been received, received packet data, packet data to be transmitted, and the like.

- 20 A separate, removable SIM (not shown) can be provided as well, the SIM storing, for example, a preferred Public Land Mobile Network (PLMN) list and other subscriber-related information. The ROM is assumed, for the purposes of this invention, to store a software program enabling the MCU 120 to execute the software routines and protocols required to implement a user interface (UI), via display 140 and the numeric keypad 160, in accordance with this invention.
- 25 Although not shown, a microphone and speaker are typically provided for enabling the user to conduct voice calls in a conventional manner.

The mobile station 100 also contains a wireless section that includes a digital signal processor (DSP) 180, or equivalent high speed processor or logic, as well as a wireless transceiver that

- 30 includes a transmitter 200 and a receiver 220, both of which are coupled to an antenna 240 for communication with the network operator. At least one local oscillator (LO) 260, such as a frequency synthesizer, is provided for tuning the transceiver. Data, such as digitized voice and

packet data, is transmitted and received through the antenna 240.

The user interface is the main focus of this invention, in particular a user interface that enables the user to enter and store Indian characters using only the numeric keypad 160, and to view the entered characters on the display 140. The entry of Indian characters enables the user to compose and to respond to messages, such as SMS and similar types of messages, including instant messaging applications, as well as to enter information when connected wirelessly to the internet and other data communications networks, including intranets. In the presently preferred embodiment of this invention the entered characters are Hindi characters. However, these teachings apply as well to other Indian languages, such as Punjabi, Gujarati, Oriya, Bengali, Assamese, Telugu, Kannada, Malayalam and Tamil. Other, non-Indian, languages may also benefit from the application of this invention

The user interface includes a Hindi character editor function 130A that forms a part of the software (a stored program) in mobile station memory 130. As will be discussed in further detail below, the editor function 130A operates in conjunction with a set of rules 130B that the editor function 130A consults, for example when resolving the depression of a key that can be interpreted in more than one manner. A rendering or display engine 130C component is responsible for correctly forming images on the display 140 of the characters that make up the writing. Also referred to below is an editing or text buffer 130D wherein entered characters are stored.

It should be noted that the keypad 160 need not be a physical keypad with mechanical switches, but could be implemented as a touchscreen type of device and be displayed on a portion of the display 140. The keypad could also be located remotely and communicate with the MCU 120 through a low power RF or IR link. It should further be noted, as was mentioned above, that the keypad 160 (and/or the display 140) could be a component of a host device to which the mobile station 100 is attached or installed or inserted.

By way of introduction, Devanagari script is used to write Hindi. The Devanagari script has the following properties: (a) letters run left to right on a horizontal line, with space between each word; (b) there is no distinction between capital letters and lower case letters, as in Latin script;

(c) there are both vowels and consonants; (d) the script is phonetic, meaning that it is pronounced as it is written, and syllabic, meaning that the writing is based on consonants and independent vowels that can form a syllable; and (e) characters can combine and change shape depending on their context.

5

The Unicode range for Devanagari letters is U+0900 to U+097F.

There are 33 consonants in the Devanagari script. Each consonant represents a single consonantal sound, but may also exhibit an inherent vowel (the short vowel /a/.) In the presence
10 of the dependent vowel, the inherent vowel associated with a consonant is over-ridden by the dependent vowel. It should be noted that there are 45 Devanagari consonants defined in Unicode. The number 33 given here refers to Hindi-specific (non-nukta) consonants.

Vowels can be divided into independent and dependent vowels, also known as matras. The
15 independent vowels may stand on their own, and are used to write syllables that start a word. An independent vowel may also appear within a word (e.g., see the example /bhai:/ below.) Dependent vowels cannot stand alone, and are instead applied to a consonant or consonant cluster.

20 Fig. 2 shows the matras, or dependent vowels.

There are certain Devanagari specific characters that are now considered. The Devanagari script employs a sign, referred to as a halant (Unicode value U+094D), also known as virama and as a vowel omission sign. The character halant (virama) generally indicates the omission of vowel
25 sound where it otherwise would occur. Normally, a halant is used at the end of the word to indicate that the word ends with a consonant sound (thus omitting the inherent vowel). The Indian standard ISCII extends the application of the halant: it is used to indicate the consonant clusters where two or more consonants are to be pronounced without any intervening vowel sound. Visually, such a phenomenon is usually depicted by changing the appearance of the
30 vowel-less consonant, or by replacing a sequence of consonants by a single shape, a ligature. In systems with inadequate means of displaying the consonant clusters, a halant may be displayed as a visible character to indicate the omission of the vowel sound within the consonant cluster.

Fig. 3 shows the halant.

Modifiers indicate nasalization or aspiration. The modifiers are shown in Figs. 4A, 4B and 4C, and are the candrabindu, inserted above the character (Fig. 4A), the anusvara, inserted above the character (Fig. 4B), and the visarga, inserted beside the character (Fig. 4C).

The Devanagari sign nukta, shown in Fig. 5, is a combining mark that is used with some characters to create new characters.

10 In accordance with an aspect of the teachings of this invention, the mobile station 100 is operated so as to enable the entry of Devanagari script to compose Hindi language messages. The entry of the Hindi characters is accomplished using the conventional numeric keypad 160, and thus does not require that the mobile station hardware be redesigned or modified in any way. Instead, the software of the mobile station 100 is modified so as to interpret the user input from the
15 keypad 160 in the manner described below.

Referring to Fig. 6, in accordance with one embodiment of this invention the Hindi vowels are mapped behind the keypad numeric keys 1 and 2, and the Hindi consonants are mapped behind keys 3-9. The consonants are arranged on the keypad 160 according to the vargs, with the non-
20 varg consonants being distributed over keys 8 and 9. The last character on each key is the (Latin) number printed on the key (i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9), which for convenience are used instead of the Hindi numbers.

In accordance with another embodiment of this invention the Hindi vowels are mapped behind
25 the keypad numeric keys 2 and 3, the Hindi consonants are mapped behind keys 4-9, and the Devanagari modifiers are mapped behind key 1, including virama.

In other embodiments of this invention the vowels and consonants (as well as modifiers and the space key (0)) can be mapped on other keys. As such, these keys may be referred to hereafter
30 simply as vowel keys, consonant keys, a modifier key and a space key, without specific reference to the actual key or keys that happen to be involved in the mapping.

Note that the vowel keys map both the independent and the dependent vowels. The choice of which is meant can be established automatically by the current editing context, in view of the current content of the editing buffer 130D at the current insertion point. Note in this regard that the current insertion point within the buffer 130D can be changed by changing the location of the cursor. As the insertion point in the buffer 130D is changed, then the current editing context may change as well.

The selection of a particular character for a given keypad numeric key is done by repeated key depressions, as is shown in Fig. 7 for the example of keypad numeric key 3. By pressing a key repeatedly within a timeout period the user is enabled scroll through the characters behind the key, which can be displayed in turn on the display 140 at the current character entry location or at some other predefined location. If the user pauses after a key depression for longer than the timeout period then the currently displayed character is entered into the text, and the cursor appears to the right of the current syllable. The timeout can be terminated by pressing another key.

Fig. 8 shows the keypad 160 mapping for matras (dependent vowels) on keys 1 and 2. The editor function 130A of the software in memory 130 interprets the vowel keys so as to always produce a matra when the current character insertion point (cursor location) follows a consonant (of the relevant script, e.g., a Hindi consonant).

As an example, to write /cha:pit/ छापित, the user first presses key 4 twice for consonant /cha/ छ, key 1 once for matra /-a:/ ा, key 7 once for consonant /pa/ प, key 1 twice for matra /-i/ ि and finally key 6 once for consonant /ta/ त.

Note that the matra /-i/ is entered only after the consonant /pa/ that it follows when pronounced. The rendering engine 130C function of the software in memory 130 displays the matra at the correct position on the display 140.

Fig. 9 shows the keypad 160 mapping for independent vowels on keys 1 and 2. When the character preceding the insertion point is something other than a consonant, the editor function

130A of the software in memory 130 interprets keys 1-2 so as to produce an independent vowel.

As an example, to write /utha:u:/ उठाऊ, the user first presses key 1 five times for vowel /u/ उ, then key 5 twice for consonant /tha/ ठ followed by one depression of key 1 for matra /-a:/ 5 ा, and finally key 1 is pressed six times for vowel /u:/ ऊ.

Note that the single depression of key 1 after the consonant /tha/ produces matra /-a:/ instead of the independent vowel, because the editor function 130A intelligently assumes that a consonant is followed by a matra rather than an independent vowel. This type of decision making can be 10 accomplished by storing the set of language and syntax rules 130B that the editor function 130A consults when resolving the depression of a key that can be interpreted in more than one manner, such as a key depression that is interpreted based on the context of the current insertion point in the text buffer 130D.

15 Note further in this regard that the depressions of key 1 after matra /-a:/ produces an independent vowel /u:/ instead of a matra, because a matra can only occur after a consonant.

If it is required to produce an independent vowel after a consonant, the user may press the context shift key (for example the * key (asterisk)) before entering the vowel by pressing one

20 of vowel keys. For example, to write the word /bhai:/ भई the user first presses key 7 four times

for the consonant /bha/ भ, then key * followed by key 1 four times for the independent vowel

/i:/ ई . If the user did not press the key * after entering the consonant /bha/ the editor 130A

would instead produce the matra /-i:/ ी resulting in /bhi:/ भी .

25 Consonants inputted from the keys 3-9 are produced as full consonants. If the user needs to produce a consonant cluster (for example /p.ra/ प्र instead of /para/ पर), the user can change the previously entered consonant into a half-form by pressing the context shift key before entering the following consonant.

For example, to write /dilli:/ दिल्ली, the user first presses key 6 three times for the consonant /da/ द, then two times key 1 for matra /-i/ ि, key 8 three times for consonant /la/ ल followed by key * and three further depressions of key 8 to join another consonant /la/ into cluster /l.la/ लल. Finally, the user presses key 1 three times for the final matra /-i:/ ी.

5

Note that if the user did not press key * between the first and second consonant /la/, the result would have been a sequence of full consonants /lala/ लल instead of the desired consonant cluster /l.la/ लल.

- 10 The rendering engine 130C preferably automatically handles the ligatures such as /k.ssa/ and /t.ra/. For example, to write /ak.ss/ अक्ष the user first presses key 1 once for /a/ अ, then key 3 once for /ka/ क. The user then presses the key * followed by key 9 twice for /ssa/ ष. If the user did not press the key * between consonants /k.ssa/ and /k.ssa/ the consonants would not be joined as a cluster /k.ssa/ क्ष, but they would appear instead as a sequence of full consonants
- 15 (/kassa/ कष).

Note that regardless of how the entered characters are displayed to the user, for example whether or not ligatures are correctly displayed, the content of the text buffer 130D will contain the correct character sequence. It is preferred, however, that for the case where the rendering engine

20 130C is able to substitute consonant clusters with appropriate ligatures, these are displayed as soon as the user has entered an appropriate character sequence.

The editor 130A, in cooperation with the stored list of rules 130B, also changes the consonant /ra/ into superscript repha when it occurs at the beginning of a consonant cluster.

25

As an example, to write /karta:/ कर्ता the user first presses key 3 once for /ka/ क, then key

8 twice for /ra/ र, then the key * followed by one press of key 6 for the consonant cluster /r.ta/ र्त instead of a sequence of full consonants /rata/ रत, and finally key 1 once for matra /a:/ आ.

It should be noted that the ability or inability to display the consonant /ra/ in various contexts, such as the superscript repha that was discussed above, depends more specifically on the capabilities of the rendering engine 130C, and not the editor 130A.

Other Hindi characters can be entered as well using the editor 130A in accordance with the teachings of this invention.

10

For example, the modifiers (anusvara ँ, carabindu ँ and visarga ः) as well as nukta ॠ, can be obtained through a Special Character Screen (SCS) 140A that is opened on the display 140 by a long depression of, for example, the context shift key. The SCS 140A, which can be considered as well to be a special characters window, includes a Special Characters List that contains at least the modifiers anusvara ँ, carabindu ँ, visarga ः) and nukta ॠ.

15

As an example, to write /a:mkra:/ आँकड़ा, the user presses key 1 once for vowel /a:/ आ, then user presses key * for a long period (at least about one second) to open the SCS 140A, then navigates the Special Character List to select the carabindu ँ from the list. The user then presses key 3 once for /ka/ क. For the nukta consonant /rha/ ढ the user first presses key 5 three times for the consonant ढ and then accesses the Special Character List 140A for nukta ॠ. Finally, user presses key 1 once for the matra /-a:/ आ.

20

It should be noted that the last depression of key 1 produces the matra /-a:/, since the last entered character was a (nukta) consonant. The editor 130A, in cooperation with the rules 130B, combines the sequence of the consonant and the nukta into one single character.

25

The characters appearing in the Special Character List can be accessed by navigating the

5 selected character into the text by pressing key 5. In this case the Special Character Screen 140A remains open, and the user can then continue by selecting another character from the Special Character Screen 140A.

map the modifiers and other signs directly to one of the keys of the keypad 160. As an example, Devanagari modifiers and other signs can be directly mapped to key 1, the vowels can be mapped to keys 2 and 3, and the consonants can then be mapped to keys 4, 5, 6, 7, 8 and 9. Other mappings, key selections and arrangements can also be accomplished, which should become apparent to those skilled in the art when guided by these teachings.

When the cursor is moved within the displayed text it stops only between Hindi syllables, as illustrated in Fig. 10. The thin vertical lines show the locations where the cursor can stop.

As the cursor is only allowed to stop between the Hindi syllables, it is not possible to place the
 20 cursor within a consonant cluster, or between a consonant and the independent vowel following
 it. In the example of Fig. 10 it can be seen that there are cursor stops indicated around the
 syllables व्य, लों, कि and ध्या, but not within them.

Referring to Fig. 11, the backwards deletion of the characters occurs in exactly the reverse order
25 to entering the characters. While adding the matra of the short vowel /-i/ the editor 130A
automatically places it at the left side of the consonant that it is attached to. As the deletion is
performed in exactly reverse order, the matra deleted before the consonant.

It should again be noted that the encoding of the Hindi text happens in strictly logical order. For
 30 example, when the matra short /-i/ is added after a consonant, it is stored in the editing buffer
 130D directly after the consonant. However, according to the rendering rules of the Devanagari
 script the matra short /-i/ visually appears at the left side of the entered consonant. This behavior

is handled by the display engine 130C, which may also consult the set of rules 130B or a separate set of rendering or visualization rules, and can be essentially independent of the editing application performed by the editor function 130A.

5 It can be realized that the foregoing process of selecting and displaying Hindi characters normally includes buffering the selected characters (actually codes representing the selected characters) into the text buffer 130D in the memory 130, and then subsequently transmitting the buffered characters in a message or in a packet to the wireless network through the transmitter 210.

10

While described in the context of specific keys, specific character mappings and the like, it should be realized that these are exemplary of this invention and represent a presently preferred embodiment. However, in other embodiments of this invention other keys and character mappings could be used. As but one example, Figs. 6, 8 and 9 show the vowels mapped on keys 15 1 and 2 in strict alphabetical order. However, in other embodiments other mappings could be used, such as by placing the long vowels on key 1 and the short vowels on key 2, or vice versa. The vowels can also be placed on other keys, such as keys 2 and 3, as was discussed above.

Thus, this invention is not intended to be limited to only the embodiments disclosed herein, but 20 is to be given a wider scope of interpretation consistent with the appended claims.